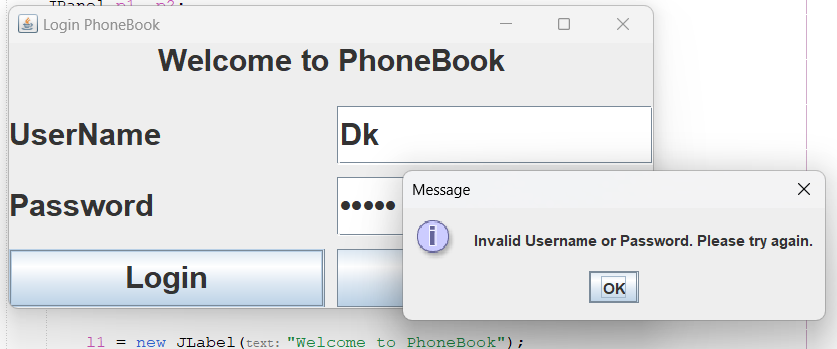
Phonebook Application

Data structure algorithm

|  |  |
| --- | --- |
| Members | Student number |
| Tangeni Shalauda (GL) | 223077623 |
| Kleopas Andima | 223064211 |
| Deciderius Katopa | 223071455 |
| Phillann Genuine Shitaleni | 223113778 |
| Barnabas Iihuhwa | 223081701 |
| Otto Mwandingi Kapia | 223123382 |

Submission Date: October 9,2024

Phonebook Management Screen shorts



A screenshot of a computer

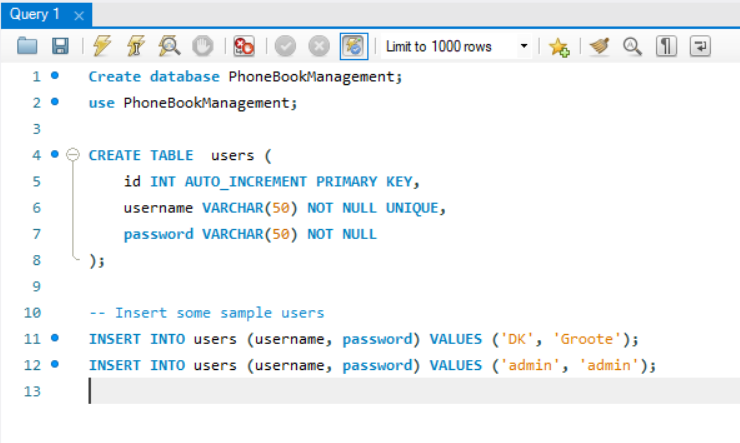
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



Section A: Algorithm representation (Pseudocode)

= The Phonebook Application is designed for a Namibian telecommunications company to efficiently manage contacts using linear data structures. The application implements essential functionalities like inserting, searching, deleting, updating, and displaying contacts.

Pseudocode

1. Insert Contact

FUNCTION InsertContact(name, number)

IF name or number is empty THEN

DISPLAY "Please enter both name and number."

RETURN

END IF

ADD new Contact(name, number) to phonebook

CALL DisplayContacts()

END FUNCTION

2. Search Contact

FUNCTION SearchContact(name)

IF name is empty THEN

DISPLAY "Please enter a name to search."

RETURN

END IF

FOR EACH contact IN phonebook DO

IF contact.name equals name THEN

DISPLAY "Contact found: " + contact

RETURN

END IF

END FOR

DISPLAY "Contact not found."

END FUNCTION

3. Display All Contacts

FUNCTION DisplayContacts()

IF phonebook is empty THEN

DISPLAY "No contacts available."

RETURN

END IF

DISPLAY "Phonebook Contacts:"

FOR EACH contact IN phonebook DO

DISPLAY contact

END FOR

END FUNCTION

4. Delete Contact

FUNCTION DeleteContact(name)

IF name is empty THEN

DISPLAY "Please enter a name to delete."

RETURN

END IF

FOR EACH contact IN phonebook DO

IF contact.name equals name THEN

REMOVE contact FROM phonebook

DISPLAY "Contact deleted."

RETURN

END IF

END FOR

DISPLAY "Contact not found."

END FUNCTION

5. Update Contact

FUNCTION UpdateContact(name, newNumber)

IF name or newNumber is empty THEN

DISPLAY "Please enter both name and new number."

RETURN

END IF

FOR EACH contact IN phonebook DO

IF contact.name equals name THEN

contact.number = newNumber

DISPLAY "Contact updated."

RETURN

END IF

END FOR

DISPLAY "Contact not found."

END FUNCTION

6. Sort Contacts (Optional)

FUNCTION SortContacts()

SORT phonebook by contact.name

DISPLAY "Contacts sorted."

END FUNCTION

7. Efficiency Analysis

FUNCTION AnalyzeSearchEfficiency(name)

START time measurement

CALL SearchContact(name)

END time measurement

DISPLAY "Search time measured."

END FUNCTION

Section B: Practical Implementation

Overview of the Implementation

The Phonebook Application is developed in Java using the Swing library for the user interface. It utilizes an ArrayList to manage the contacts. Users can perform operations like inserting, searching, updating, deleting, and displaying contacts through a graphical interface.

Code Description

UI Initialization (initUI method): Initializes the GUI components such as text fields and buttons.

Event Handling: Each button has an action listener that triggers the corresponding operation (e.g., Insert, Search).

Functional Methods: These methods implement the core functionalities of the application:

Insert Contact: Adds a new contact.

Search Contact: Finds a contact by name.

Delete Contact: Removes a contact from the phonebook.

Update Contact: Modifies an existing contact's number.

Sort Contacts: Sorts contacts alphabetically.

Display Contacts: Shows all contacts in the text area.

Efficiency Analysis: The search operation's efficiency is measured in nanoseconds, displaying the duration for search queries

The java code

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

public class Phonebook extends JFrame {

private ArrayList<Contact> phonebook;

private JTextArea displayArea;

private JTextField nameField, numberField;

private JButton insertButton, searchButton, deleteButton, updateButton, sortButton,displayButton;

public Phonebook() {

phonebook = new ArrayList<>();

initUI();

}

private void initUI() {

// Panel for input fields and buttons

JPanel panel = new JPanel(new GridLayout(3, 2));

JLabel nameLabel = new JLabel("Name:");

nameField = new JTextField();

JLabel numberLabel = new JLabel("Number:");

numberField = new JTextField();

panel.add(nameLabel);

panel.add(nameField);

panel.add(numberLabel);

panel.add(numberField);

// Buttons for operations

insertButton = new JButton("Insert");

searchButton = new JButton("Search");

deleteButton = new JButton("Delete");

updateButton = new JButton("Update");

sortButton = new JButton("Sort");

displayButton =new JButton("Display All");

panel.add(insertButton);

panel.add(searchButton);

panel.add(deleteButton);

panel.add(updateButton);

panel.add(sortButton);

panel.add(displayButton);

// Text area to display contacts

displayArea = new JTextArea(10, 30);

JScrollPane scrollPane = new JScrollPane(displayArea);

// Adding components to frame

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

// Button action listeners

insertButton.addActionListener(e -> insertContact());

searchButton.addActionListener(e -> searchContact());

deleteButton.addActionListener(e -> deleteContact());

updateButton.addActionListener(e -> updateContact());

sortButton.addActionListener(e -> sortContacts());

displayButton.addActionListener(e -> displayContacts());

setTitle("Phonebook Application");

setSize(400, 400);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

}

private void insertContact() {

String name = nameField.getText();

String number = numberField.getText();

if (name.isEmpty() || number.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please enter both name and number.");

return;

}

phonebook.add(new Contact(name, number));

displayContacts();

clearFields();

}

private void searchContact() {

String name = nameField.getText();

if (name.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please enter a name to search.");

return;

}

for (Contact contact : phonebook) {

if (contact.getName().equalsIgnoreCase(name)) {

displayArea.setText("Contact found: \n" + contact);

return;

}

}

displayArea.setText("Contact not found.");

}

private void deleteContact() {

String name = nameField.getText();

if (name.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please enter a name to delete.");

return;

}

for (Contact contact : phonebook) {

if (contact.getName().equalsIgnoreCase(name)) {

phonebook.remove(contact);

displayContacts();

clearFields();

return;

}

}

displayArea.setText("Contact not found.");

}

private void updateContact() {

String name = nameField.getText();

String newNumber = numberField.getText();

if (name.isEmpty() || newNumber.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please enter both name and new number.");

return;

}

for (Contact contact : phonebook) {

if (contact.getName().equalsIgnoreCase(name)) {

contact.setNumber(newNumber);

displayContacts();

clearFields();

return;

}

}

displayArea.setText("Contact not found.");

}

private void sortContacts() {

Collections.sort(phonebook, Comparator.comparing(Contact::getName));

displayContacts();

}

private void displayContacts() {

displayArea.setText("Phonebook Contacts:\n");

for (Contact contact : phonebook) {

displayArea.append(contact.toString() + "\n");

}

}

private void clearFields() {

nameField.setText("");

numberField.setText("");

}

// Efficiency Analysis for Search Operation

private void analyzeSearchEfficiency(String name) {

long startTime = System.nanoTime();

for (Contact contact : phonebook) {

if (contact.getName().equalsIgnoreCase(name)) {

break;

}

}

long endTime = System.nanoTime();

long duration = endTime - startTime;

System.out.println("Search time (ns): " + duration);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new Phonebook().setVisible(true));

}

// Contact class definition

class Contact {

private String name;

private String number;

public Contact(String name, String number) {

this.name = name;

this.number = number;

}

public String getName() {

return name;

}

public String getNumber() {

return number;

}

public void setNumber(String number) {

this.number = number;

}

@Override

public String toString() {

return name + ": " + number;

}

}

}

Flowchart

A screenshot of a computer screen

Description automatically generated